NASA-CR-201411

KATE: From the Lab to the Firing Room

Charles Pepe
Steve Beltz
Robert Merchant
Steve Nunez
Charles Goodrich
I-NET, Inc., M/S INI-16, Kennedy Space Center, FL 32899

Abstract

For the past 9 years, the AI group at NASA's Kennedy Space Center (KSC) has been developing a real-time model-based reasoning (MBR) system that can diagnose faults and provide control advisories. The system resulting from this effort is called KATE (Knowledge-based Autonomous Test Engineer). This article describes the transition of KATE from the applied research lab to the firing room (the firing room is the main control and monitoring center for all Shuttle ground support and launch processing). Because the problems that were encountered along the way are not unique, we decided to share some of our experiences. This article also includes a brief overview of the latest application of KATE.

INTRODUCTION

During the ground processing and launch countdowns of the Space Shuttle at the Kennedy Space Center, NASA relies on one of the world's largest automated process control systems known as the Launch Processing System (LPS). This system works well under normal operating conditions, but is often inadequate when anomalies occur. Like conventional control software, LPS is only able to detect and correct anomalies that the program authors anticipated and is programmed to "hold" the countdown if unanticipated sensor readings are encountered.

Since 1983, NASA has been developing and testing various AI techniques intended to improve existing ground launch monitoring capabilities. Recent efforts involved testing the KATE model-based reasoning system against various test fixtures, improving its capabilities, and finally working to deliver KATE to a customer. Successful deployment required us to maintain existing KATE functionality while porting that functionality to a customer-acceptable hardware/software platform. This paper gives a brief history of the KATE project and describes the changes that were necessary to make KATE a viable production system.

KATE in LISP

NASA's first success in applying MBR technology came in 1985 with the development of LES (Liquid Oxygen Expert System). LES modeled a part of the Shuttle Liquid Oxygen (LOX) Tanking system, concentrating on discrete measurements and system generalities. Though the representation of the domain was simple, LES did prove successful enough to warrant a follow-on project. The new system, KATE, extended the LES algorithms and included control capabilities. This early version of KATE was tested with a scale model of the Environmental Control System (ECS) of the Orbiter Modification and Refurbishment Facility. KATE successfully demonstrated its ability to diagnose and control using the ECS equipment.

In late 1988, a major rewrite of KATE was undertaken to incorporate a new knowledge representation and expand its diagnosing and control capabilities. This version of KATE took 12 months to complete and was developed in LISP on a Symbolics LISP Machine. There were approximately 70k lines of code that included a new knowledge base (KB) representation (the structure and function of components were separated), new algorithms for diagnosing and control, and a new graphical user interface (GUI). The new GUI featured plotting, a semi-automatic drawing system (for advanced electrical schematics with live data), animated icons (e.g., storage tanks filling up, valves opening and closing), and a user interface building tool.

KATE LISP Applications

In the fall of 1989, two applications were chosen to be built using this new version of KATE. The first was a joint effort between NASA and the USAF for the Advanced Launch Operations (ALO) of the Advanced Launch System. To prove the viability of KATE for a new launch system, they directed us to demonstrate KATE with actual hardware. We chose to build a model that closely resembled a space vehicle tanking system, using water instead of a cryogenic liquid. The goal was to build a KATE application to perform autonomous monitoring, diagnosis and control of the hardware. Twelve months later KATE was successfully demonstrated using a KB that contained about 250 objects. The KATE ALO demo has been presented over 30 times and continues to be a valuable system for proving the feasibility of advanced software in the launch environment.

The second test scheduled for this new KATE system was another demonstration of the Shuttle LOX tanking system. Since the LES system had successfully monitored a part of the LOX system, the hope was that this new KATE could be used to monitor the entire system. Though only a prototype, the KATE LOX application was designed to diagnose actual component failures as they happened and recommend corrective actions. Control of the hardware was not required (though KATE did keep track of what phase the system was in and the time remaining in that phase). After six months of development, KATE LOX began to monitor actual LOX tankings; after another six months of KB development, the LOX KB grew to include 2500 objects. The system successfully diagnosed several actual component failures (including a tachometer and a flow meter) before the LPS detected any anomaly and before the console operator determined what failed. The KATE LOX system continues to monitor Shuttle LOX tankings.

Unfortunately, there was a flaw in our development process. We produced a large amount of code and several complicated KB's in a relatively short period, but the systems were developed with no structured methodology. The resulting program had no clear system design, no modularity, too much interdependence of data structures, and inadequate documentation. The KATE LISP system was powerful and capable of effective demonstrations, but often required a KATE expert for proper operation.

The success of KATE LISP was encouraging; however, since it was not a deliverable product, the KATE development team was presented with a new problem: how to transform a very powerful demonstration system into a useable delivery system.

Translate or Reimplement

We realized at this point that the KATE system needed to be reengineered before it would be accepted as anything more than a prototype. Since LISP was unacceptable as a delivery language to our user community, there were numerous discussions about whether it was feasible to translate the LISP code. We considered using a tool that attempts to translate LISP to C automatically. There were two major problems with this approach. First, the KATE code was not well designed from a system level. A translation without a redesign would not fix the problems. Second, if a translation tool were used the code generated would be unmaintainable. The system would have to be maintained in LISP and delivered in translated C. Because of these reasons we decided to reengineer the system rather than translate it. We also decided to rename the system KATE-C.

SOFTWARE ENGINEERING APPROACH

Before the AI group could turn KATE into production quality software for the firing room, it was essential for the team to adopt a structured software engineering paradigm instead of the rapid prototype style that was used in KATE LISP. To help us, we used a development methodology that included the following guidelines:

1. Develop formal requirements
2. Build a design to meet the requirements
3. Implement the design
4. Test code against requirements
5. Use technical reviews
6. Use programming guidelines
7. Use documentation standards
8. Exploit object oriented technology (code reuse, polymorphism, encapsulation, etc.)

It should be noted that requirements, design and implementation were not always performed in order. As each module neared completion, there was a natural tightening of the requirements, design and code.

KATE in C++

The first big hurdle was deciding on the platform, operating system, and language. Getting KATE into the hands of users required understanding their working environment, what hardware they were familiar with, and what type of system would be cost effective for them. We also needed to address the technical issue of what OS/language would be powerful enough to handle the KATE reasoning mechanism. After some thought it was decided the best delivery platform would be a UNIX workstation using C++ as the development language and X Windows and Motif for the user interface (UI). The user community was beginning to use UNIX workstations and though C was more prevalent than C++, we felt that KATE required the object oriented design (OOD) paradigm if it was going to be successful. And though the group had little X/Motif experience, it was clearly becoming the industry standard for user interfaces on UNIX platforms.

SYSTEM DESIGN

We took a two-part approach to system design. We decided to functionally decompose the system into high level modules and then apply an object oriented design to those modules. The one design constraint that we imposed was to stay consistent with the architecture of CCMS-2 (the new hardware/software design of the Firing Room). This constraint required a client/server design so KATE-C could operate in a distributed application processor/display processor environment.

The functional decomposition resulted in three top-level modules: the User Interface, the Reasoning Module and the Data Module. Each module would run in its own process and the processes would communicate using an interprocess communication system.

As we began the process of reengineering KATE, we developed a few high-level goals that would help the system be accepted by our user community:

1. Deemphasize autonomy
2. Emphasize advisory
3. Involve the user in KATE-C's reasoning process
4. Make the software as fault tolerant as possible

Below are some design decisions and techniques that proved useful in the reengineering effort.

User Interface Module

The KATE-C GUI was designed with only a few assumptions about the user's applications. The goal was to design an

Application Program Interface (API) that the customer could use to implement their own, application-specific, user interface.

The GUI provides a basic framework upon which the customer can build. This framework provides all of the services necessary to interact with the other KATE-C modules such as the X Window event loop and message handling. The programmer need only know the X/Motif system to develop a custom GUI. Also, associated with each module or submodule was a text UI. The text user interfaces were tremendously helpful; developers could test their code and new messages without running the GUI.

Reasoning Module

As we started to design the reasoning mechanisms for KATE-C, our main goal was to produce a system that was functionally equivalent to the LISP system. By applying object-oriented analysis and design techniques, the resultant module turned out to be as powerful as the LISP system and the design was clear, maintainable, and extendable.

The design is centered around a Reasoner object. The Reasoner object is the main controlling agent for the four top-level objects: monitoring, simulation, fault detection, and diagnosis. In addition, each component in the knowledge base has as an inherited set of member functions and data members that determines how that particular object shall function with respect to the four reasoning tasks.

There were four key changes that occurred during the re-engineering effort:

First, we decided to separate the major reasoning functions into stand alone submodules. This straight-forward idea greatly improved the design by allowing for better encapsulation and easier testing.

Second, we elevated the task of determining if a sensor is discrepant to an independent reasoning task and called it Fault Detection. The object-oriented design of fault detection greatly increased the system's ability to reliably determine if discrepancies exist. Previous KATE implementations treated all measurements the same. One function was applied to all measurements to determine if they were discrepant. The OOD of the new Fault Detector gives individual measurements the responsibility to determine if they are discrepant. Noisy sensors now can be evaluated differently than highly accurate sensors.

Third, we implemented a Diagnoser Tool Box. The diagnoser tool box helped turn a very complicated LISP algorithm into several smaller and simpler diagnostic tools. In addition, the tool box approach provides a mechanism for adding general tools (i.e., diagnosing over time tool) and domain specific tools (i.e., a loss of cooling tool for a Freon cooling system).

The fourth change that significantly improved the KATE-C system resulted from two high-level goals that we had for the system: deemphasizing autonomy and emphasizing advisory. The KATE-C system was designed to bring the user into the reasoning loop by providing advisory reports on what may have failed in the system and providing tools to help the user confirm or discard the reports. With this approach the system is less threatening to the users and the users are more accepting of the system if it incorrectly reports an anomaly.

Data Module

The Data Module is the other top-level module in KATE-C. The Data Module's main purpose is to provide data to the Reasoning Module. Though conceptually simple, this module also greatly benefited from the OOD paradigm. New data providers can be added as plug-in components with little or no change to the supporting code.

Knowledge Representation

A KATE-C knowledge base is a collection of objects that can be used by KATE-C's reasoning algorithms to simulate both the behavior of individual components in the hardware and the overall system behavior. A knowledge base is built by selecting the appropriate objects from a library and specifying how those components are connected. Each object in the library has as part of its description a set of inputs and one transfer function. The inputs enable objects to be connected and the transfer function takes the input values and generates an output value. Each library object is implemented as a C++ class which evolved directly from the KATE LISP frame representation.

Every object in a KB is either a component, pseudo object, or a function designator (FD). Component objects represent actual system components that are subject to faults such as valves, pumps, circuit breakers, etc. FD's are the actual commands and measurements that are used to control and monitor the equipment. All objects in a KB that are not either components or FD's are called pseudo objects. Pseudo objects can represent physical quantities such as flow rates and temperatures and they also can be used to connect components that are functionally related but are not structurally connected.

The transition from LISP frames to C++ classes was straight forward and the resulting C++ representation was more concise. We were able to represent the same information in less code.

THE VEHICLE HEALTH MANAGEMENT SYSTEM

While KATE-C was being developed it was chosen to be the basis for a system to be developed by the LPS Application Software group of Lockheed Space Operations Company at KSC called the Vehicle Health Management System (VHMS). The VHMS project involves using the KATE-C model-based reasoning technology to build a system that will reduce the cost of processing the Shuttle.

During normal ground processing the Shuttle is powered up and system engineers are required to monitor their respective vehicle subsystems from consoles in the firing room. In addition, teams of engineers monitor those same subsystems at the Orbiter Integration Console.

The objective of the VHMS project is to provide an intelligent monitoring/diagnosis system for the Orbiter Integration Console. The subsystems to be monitored by VHMS include the Environmental Control and Life Support Systems (ECLSS), Electrical Power Distribution and Control (EPDC), and the

Instrumentation (ISL). Currently, each system requires at least one engineer to be on console when the Shuttle is powered up, even when no tests are being conducted with that subsystem.

If VHMS is successful, its use at the Integration Console will free the system engineers from "baby-sitting" the Shuttle and allow them to do more productive work. This change would result in significant cost savings per orbiter processing flow.

As a KATE-C application, the VHMS system will include a customized user interface built within the KATE-C GUI framework, an application knowledge base, and the KATE-C Reasoning and Data Modules.

VHMS Domain

The initial Shuttle system chosen for development in VHMS was the Freon Coolant Loop (FCL) subsystem of the Active Thermal Control System (ATCS). The ATCS is a subsystem of ECLSS. The FCL subsystem consists of two nearly identical loops located in the orbiter, each of which flows Freon-21. The Freon is circulated by one of two redundant pumps. The heat from active equipment is absorbed by the Freon and then transferred to the ground support equipment (GSE) heat exchanger. The heat exchanger acts as a heat sink for the ATCS when the Shuttle is powered up on the ground. Since critical onboard computer equipment and instrumentation can be damaged by overheating, the FCL temperatures and flow rates must be monitored very closely for loss-of-cooling conditions.

VHMS Knowledge Base

The purpose of the initial VHMS KB is to model the dynamic temperatures, pressures, flow rates, and component states of the ATCS so that VHMS can detect and diagnose component faults when they occur. Eventually the KB will be expanded to include the Shuttle's Atmospheric Revitalization System (ARS) and EPDC subsystems as well.

The FCL KB took approximately 6-8 months to develop. Heat transfer is a difficult process to model quantitatively due to the fidelity that is required to detect and diagnose faults. To model the Freon loops properly, a substantial part of the electrical power distribution and ground systems also had to be modeled.

There are approximately 50 different classes of objects in the KB, and about 400 instance objects. About 60 percent of the object classes represent actual components, while the remaining 40 percent represent pseudo objects (i.e., system parameters such as temperatures and flow rates). In contrast to the class percentages, about 75 percent of the instance objects represent components, and 25 percent represent pseudo objects. The reason for this difference is that the number of class instantiations is not the same for components and pseudo objects. A typical component class will have several instances of itself in the system, whereas most pseudo objects are unique.

The knowledge base was developed jointly by I-NET and Lockheed personnel. I-NET computer scientists supplied the KATE-C (and model building) expertise, while Lockheed programmers and Shuttle engineers contributed the domain expert knowledge. Lockheed also designed a set of customized application windows that function as subsystem display panels, providing the console (monitoring) engineer with a quick status at a glance of several ECL subsystems.

VHMS Goals

A big concern of the engineers at the Integration Console is failure of a main DC power bus in the orbiter's EPDC system. A main bus drop can cause the Integration Console to become swamped with fault conditions. In certain circumstances, a loss-of-cooling condition could occur, placing critical reaction time limits (usually 2 to 5 minutes) on the console engineers. The console engineers must reconfigure the system so that Freon cooling can resume. If the hardware cannot be reconfigured within the specified time limits, the orbiter must be powered down, a costly and dangerous procedure. VHMS's role is to notify the console operator when a bus failure has occurred, identify which bus, and if there is a potential for a loss-of-cooling condition, determine what actions should be taken and the time constraints.

Another goal for VHMS is to distinguish sensor failures from component faults. There is not a lot of redundancy in the FCL system, and often it is difficult (or time consuming) to determine if an anomaly is a sensor problem or a component problem. It is hoped that the console engineer could verify sensor failures quickly with the VHMS system.

SUMMARY

After many years of building KATE prototypes and proof-of-concept demonstrations, the KATE-C system is now ready for deployment. The reengineering effort resulted not only in producing a well designed and maintainable KATE-C system, but also taught the development team the value of a good development plan, the need for a structured development process, and the power of the object-oriented design paradigm. The VHMS system is tentatively scheduled for deployment in the firing room in 1996.

Acknowledgements

A lot of people contributed to the KATE-C and VHMS development effort. We would like to thank: Mark Schnitzius, Scott Budzowski, RJ Edwards, Greg Hadaller, Richard Owens, Chris Walker, Dave Chenault, Barbara Brown, Peter Engrand, Dave Clark, Rich Ikerd, and Barbara Kerschner.

References

Davis, R. 1984. "Diagnostic Reasoning Based on Structure and Behavior." *Artificial Intelligence* 24, (Dec.): 347-410

Jamieson, J., E. Scarl, and C. I. Delaune 1985. "A Knowlege Based Expert System for Propellant System Monitoring at the Kennedy Space Center." In *Proceeding of the 22nd Space Congress* (Cocoa Beach, FL. Apr.) 1-9

Pepe, C., et al 1992. "KATE - System Overview and Project Description", NASA Report

Scarl, E., J. Jamieson, E. New, 1988. "Model-Based Reasoning for Diagnosis and Control", FLAIRS